TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Interaktivní webový knihovní systém

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika

Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Ondřej Tůma**

Vedoucí práce: prof. Ing. Zdeněk Plíva, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Interactive web-based library system

Diploma thesis

Study programme: N2612 – Electrotechnology and informatics

Study branch: 1802T007 – Information technology

Author: **Bc. Ondřej Tůma**

Supervisor: prof. Ing. Zdeněk Plíva, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Tůma**
Osobní číslo:	**M12000236**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Interaktivní webový knihovní systém**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s požadavky na ústavní knihovní systém a s moderními technologiemi databází, např. architekturou rozhraní REST
2. Sestavte základní strukturu databáze a vytvořte jádro aplikace pro správu ústavní literatury jako webovou službu
3. Vytvořte klienty pro platformu PC a mobilní zařízení s využitím technologie QR-kódů a ověřte jejich funkce na reálných datech

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 40 - 50 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **M. Pilgrim: Dive Into HTML5 (http://diveintohtml5.info/)**

[2] **R. Fielding: disertační práce (http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm)**

Vedoucí diplomové práce: **prof. Ing. Zdeněk Plíva, Ph.D.**
Ústav informačních technologií a elektroniky

Konzultant diplomové práce: **Ing. Jiří Jeníček, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání diplomové práce: **12. září 2014**

Termín odevzdání diplomové práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 9.9.2015

Podpis:

Abstrakt

Diplomová práce se zabývá problematikou knihovního systému řešenou za pomocí moderních technologií v oblasti webových aplikací. V první části jsou rozebrány možnosti použití technologií a jejich porovnání. V druhé části jsou popsány vybrané technologie pro tvorbu knihovního systému. Třetí část diplomové práce pojednává o přípravě linuxového serveru a instalací podpůrných aplikací, které jsou potřeba pro provoz knihovního systému. Čtvrtá kapitola obsahuje návrh a implementaci serverové části, klientské části aplikace a klientské webové aplikace pro mobilní telefony s operačním systémem Android. V rámci diplomové práce vznikla moderní modulární webová aplikace, která slouží v rámci knihovního systému pro fakultu ITE.

Abstract

The diploma thesis deals with the library system designed using advanced technologies in the field of Web applications. In the first part dealt with the possibilities of using technology and their comparison. In the second part describes the selected technology for creating a library system. The third part of the thesis deals with the preparation of the Linux server and installing supporting applications that are needed for the operation of the library system. The fourth chapter contains the design and implementation of server, client applications and client Web application for mobile phones with the Android operating system. Within the thesis originated the modern modular Web application, which is used within the library system for the Faculty of ITE.

Poděkování

Chtěl bych poděkovat všem, kteří mi pomohli při tvoření diplomové práce. Zejména pak panu prof. Ing. Zdeňku Plívovi za bezmeznou ochotu kdykoliv pomoci a poradit. Dále pak své manželce za její pochopení pro danou věc a testování aplikace.

Obsah

Seznam zkratek

API	Application Programming Interface
CGI	Common Gateway Interface
CPAN	Comprehensive Perl Archive Network
DOM	document object model
GUI	grafické uživatelské rozhraní
HTML	HyperText Markup Language
HTML5	HyperText Markup Language version 5
HTTP	Hypertext Transfer Protocol
HW	hardware
LAMP	Je zkratka, která v informatice označuje sadu svobodného softwaru používaného jako platforma pro implementaci dynamických webových stránek.
MPM	MultiProcessing module
MVC	Model-View-Controller
OS	operační systém
PHP	Hypertext Preprocessor
REST	Representational State Transfer
SQL	Structured Query Language
SPA	single-page application
SSL	Secure Sockets Layer protokol poskytující zabezpečenou komunikaci
SW	software
TLS	Transport Layer Security kryptografický protokol, poskytující zabezpečenou komunikaci
URL	Uniform Resource Locator je řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na internetu

Seznam obrázků

1 Úvod

Tématem této diplomové práce je tvorba informativního webového knihovního systému za použití moderních technologií. Jednoduchý knihovní systém zná téměř každý člověk. Každému se jistě vybaví návštěva knihovny, kdy po vybrání knížek přišel k pultu. Za pultem již vyčkával personál, který po předložení kartičky ochotně přepsal do systému jaké knížky si hodlá návštěvník knihovny vypůjčit.

V dnešní uspěchané době je již tento proces půjčování knih a jiných publikací velmi pomalý. V diplomové práci jsou srozumitelně popsané možnosti dnešních webových technologií, které za pomocí již běžného zařízení v počítači nebo v mobilním telefonu dokáží vytvořit základ pro webovou aplikaci.

Avšak webové technologie v posledních letech se těší velké oblibě a díky tomu lze dnes nalézt velké množství způsobů a technologií, které lze použít pro vytvoření interaktivního webového knihovního systému. Diplomová práce obsahuje seznámení s nejčastějšími volenými způsoby technologií. Z nichž na základě porovnání vyberu jednu a pomocí ní bude v dalších kapitolách práce popsán postup tvorby webové aplikace od návrhu po hotové řešení.

2 Cíle práce

2.1 Analýza zadání práce

Cílem diplomové práce je vytvoření moderního interaktivního webového knihovního systému pro potřeby Ústavu informačních technologií a elektroniky ITE na Technické univerzitě v Liberci. Mělo by se jednat o otevřený systém, který umožní jednak prohlížení celé databáze, standardní obsluhu pověřenou osobou „knihovníka“, ale také aktivní přístup jednotlivých zaměstnanců při půjčování a vracení vybraných publikací. Obsluha systému by měla být založena na běžně dostupných platformách bez nutnosti složitého instalování, s co největším uživatelským komfortem, ale při zachování základní úlohy takového systému, tedy zachování přehledu o jednotlivých položkách databáze. Těmito položkami mohou být tradiční knihy, ale též bakalářské a diplomové práce ústavu, CD, DVD atp.Celý systém by měl být vytvořen tak, aby byl snadno přenositelný a použitelný obecně, tedy i mimo konkrétní pracoviště a zadání.

2.2 Stávající knihovní systémy

Úloha implementace výpůjčního knihovního systému patří mezi základní příklady při studiu databází a tak je nutno konstatovat že existuje celá řada řešení této problematiky. Tyto jednodušší aplikace jsou však založeny na samotné obsluze implementované databáze s jedinou obsluhou v pozici knihovníka. Existují i různě rozsáhlé systémy, které však vesměs nenabízí využití moderních mobilních technologií a pozice „čtenáře“ je u všech těchto systémů založena na pasivním nahlížení

do databáze a komunikací s „knihovníkem“. Jestliže porovnáme dostupná řešení, lze knihovní systémy z pohledu obsluhy databáze rozlišit na 3 skupiny:

- Program, který uchovává celou databázi v lokálním souboru nebo vně sebe sama.
- Software, který má architekturu klient-server, kde klient je napsaný jako aplikace a je nutná její instalace na pracovní stanici. Velice často je daný software závislý na operačním systému, případně i na architektuře procesoru.
- Webová aplikace s databází na serveru.V mnoha případech nezávislá na operačním systému a architektury procesoru. S možností využití všech zařízení, které dokáží zobrazit webovou aplikaci (v dnešní době to je většina telefonů, tabletů a jejich derivátů).

1. skupina

Systémů pro evidenci a správu „skladových položek“ (za něž lze považovat i knihy, filmy a podobná média tohoto typu) je velmi mnoho. Například se jedná o program, který je vytvořen pro konkrétního uživatele a je tedy plně přizpůsoben místu, kde je nasazen s minimální možností distribuce dalším uživatelům. Jako příklad lze uvést program s název Evidence LSOFT[26]. Software byl vyvinut českou firmou a jedná se o plně komerční záležitost, bez možnosti využití pro menší nasazení - programy tohoto typu nejsou ideální pro úlohy jež jsou zahrnuty v cílech této práce.

2. skupina

Druhou skupinu tvoří systémy vhodné a projektované pro nasazení ve větších organizacích či knihovnách. Jsou to jednoúčelové aplikace a programy napsané přímo pro městské a jiné knihovny; jako takové splňují většinu požadavků knihoven na evidenci pohybu či umístění jednotlivých položek. Zástupcem této skupiny je mimo jiné i program Evilit od české firmy NORMSERVIS s.r.o.[27]. Nevýhodou této

skupiny je často závislost na jediné platformě a omezená modularizace celého systému, či problematická přizpůsobitelnost programu danému prostředí. Ve smyslu zadání práce se jako nevýhoda jeví pevně definovaná role knihovníka, bez možnosti aktivního přístupu běžného uživatele.

3. skupina

Poslední skupinu tvoří systémy založené na webových aplikacích s databází umístěnou na vhodném serveru; s touto databází pak jednotliví uživatelé pracují ze svých zařízení a toto řešení se jeví jako ideální platforma a architektura pro splnění zadání práce. Je možno vyhledat celou řadu aplikací, které již byly vytvořeny ať za pomoci PHP či jiného skriptovacího či programovacího jazyka; tyto aplikace jsou i volně použitelné, bohužel však převážnou většinu řešení tvoří jednoduchá webová aplikace, která ve smysli zadání práce využití. Specializované knihovní systémy tvoří jen malou část těchto aplikací a obvykle jsou implementovány do dalších systémů a není možné pouze vyjmout část kódu a použít pro naše zadání. Většina takto navržených aplikací umožňuje klientovi pouze prohlížení daných publikací a zjištění jejich stavu. Úprava těchto aplikací je obtížná nebo dost často nereálná z důvodu ochrany zdrojových kódů.

Příkladem pro tuto skupinu nechť je webový portál Portaro[28], který je provozován univerzitní knihovnou TUL. Webový portál obsahuje nejen mnoho užitečných informací o univerzitní knihovně, ale obsahuje řadu funkcí, které se týkají publikací. Jako nejvýznamnější bych uvedl katalog univerzitních publikací, které byly digitalizovány. V něm lze nalézt a následně prohlédnout digitalizované publikace vzniklé jak na akademické půdě TUL, tak mimo ni. Mezi dalšími praktickými funkcemi je možnost vyhledávání publikací ve veřejných databázích a nebo burzovní portál knih a script.

Portál bohužel není stejnorodý a šlo by říci, že každou funkci portálu vyvíjel snad někdo jiný. Což je pro uživatele občas hodně matoucí. Portál se taktéž potýká s občasně dlouhou dobou zpracování dotazu (která není zaviněná internetovým připojením).

2.3 Cíle práce

Zadání práce vychází z odlišného požadavku na filozofii zápůjček a proto nebylo možné navázat na žádné z dosavadních řešení. Byly prostudovány vlastnosti a funkce vybraných knihovních systémů, byla provedena analýza dosavadního způsobu půjčování knih a publikací v knihovně ITE a jako klíčové vlastnosti vyvíjeného systému byly stanoveny bylo tyto charakteristiky:

- Jednoduchá obslužnost jak ze strany uživatele, tak administrativního pracovníka.
- Možnost uživatelské zápůjčky bez asistence administrativního pracovníka.
- Snadná implementace stávajících publikací a nových publikací do knihovního systému.
- Vypůjčení publikace z jakéhokoliv místa na univerzitě (za podmínky možnosti připojení k internetu nebo alespoň do školní sítě) .

3 Teoretická část

Pro moderní webové aplikace již téměř nikdy nestačí použití pouhého html[1] nebo html5[2] jazyka, ale je zapotřebí použití dalších technologií. V této kapitole si představíme hlavní dvě sady technologií, které jako celek lze použít velmi efektivně k vytvoření rychlé a moderní aplikace.

3.1 Představení hlavních sad

Ve výběru hlavních sad bylo myšleno taktéž na pořizovací cenu a volnosti licencí, proto nebyla zahrnuta sada od Microsoftu, ale v dalších kapitole bude právě první sada srovnávána právě s touto neuvedenou sadou. Jelikož druhá sada není závislá na výběru operačního systému.

První sada je známá již delší dobu, dalo by se říci, že téměř od počátku tvorby moderního webu. Obsah této sady lze velmi snadno zaměnit za jinou technologii, ale princip a funkčnost zůstává téměř stejný. Jeho název je LAMP. Název vznikl z použití prvních písmen použitých programů nebo programovacích jazyků.
Výčtem tedy L - Linux[3], A - Apache[4], M - MySQL[5], P -PHP[6] nebo P- Perl[12], či případně P - Python[13].

Druhá sada je poměrně mladá, ale již velmi oblíbená zejména díky flexibilního použití. Dokonce by se dalo říci, že tato sada nastoluje trend v tvorbě webu a webových aplikací. Název taktéž vznikl s použitím prvních písmen použitých jazyků a technologií. Název sady je MEAN a skládá se z M - MongoDB[7], E - Express[8], A - AngularJS[9] N - NodeJS[10].

Na následujících stránkách budou obě dvě sady detailněji popsány a v závěrečné

teoretické části bude uvedeno porovnání daných dvou sad a bude vybraná jedna z nich. Z vybrané sady bude v Praktické části vytvořen interaktivní webový knihovní systém s podporou architektury REST[11].

3.2 Představení sady LAMP

Sada LAMP byla sestavena z programů, které jsou distribuovány pod licencemi, které umožňují bezplatné užívání daných programů i v komerční sféře.

LAMP se díky tomuto faktu stal poměrně rychle velmi oblíbeným. Významnější alternativou (pomineme-li různé deriváty z kombinací programů) byla a je sada od firmy Microsoft. Pokud nebudeme brát v potaz rozdíl v použití OS, pak se obě sady lišily v HW požadavcích a v možnostech počtu obsloužení na daném HW. LAMP nabídnul za nulovou cenu (nepočítá se cena za HW) možnost zvládnout obsloužit více požadavků na server než alternativa od Microsoftu. Už jen možnost spravovat server bez použití GUI je pro LAMP velkou výhodou (i když s příchodem Microsoft server 2013 lze taky již spravovat server bez GUI).

Existuje několik projektů, které sjednotili správu LAMPu tak, že i člověk, který nemá technické znalosti pro správu serveru a ani není schopen komplikovanější konfiguraci z LAMPu je schopný zprovoznit tuto sadu pomocí jednoduchého instalátoru. Po nainstalování lze již začít s vývojem webové aplikace. Z nejznámějších bych vyzdvihnul zejména projekty WAMP[14] a XAMPP[15]. Díky nim je možno začít s vývojem projektu během několika málo minut.

3.2.1 Linux

Linux je operační systém jehož vývoj je datován od roku 1991. Otcem projektu je Linus Torvalds. Torvalds je dodnes hlavou vývoje jádra. Jádro je zveřejňováno na serveru kernel.org. Kromě něj na vývoji spolupracují tisíce programátorů z celého světa. Již delší dobu se dá říct, že vývoj jádra je z velké části placen firmami, jako je Red Hat, Intel, IBM a další.

Linux jako takový je pouze jádro operačního systému. Aby mohl být Linux po-

važován za plnohodnotný operační systém, je nutné doplnit jádro o další programy. Základ tvoří jednoduché malé programy, které označujeme jako systémové nástroje sloužící pro spuštění startu a následně i zajištění běhu systému. Krom těchto jednoduchých programů jsou i složitější a větší programy, které poskytují uživateli možnost provádět nějakou užitečnou činnost (např. Google Chrome, Atom editor). Veškeré používané nástroje i aplikace jsou volně dostupné na Internetu.

Jelikož jsou výše zmíněné nástroje i aplikace na Internetu dostupné převážně v podobě zdrojových kódů, které je nejprve nutné přeložit do formy spustitelných souborů, bylo by pro uživatele velmi nepohodlné, kdyby veškeré aplikace a programy musel sestavit pomocí kompilátorů patřičného programovacího jazyka. Proto existují distribuce, které obsahují vše potřebné v úhledném balení a někdy i v tématickém vzhledu. Každá distribuce má své vlastní pravidla a způsoby jak přidávat a instalovat nové programy a jak je konfigurovat.

Distribuce jsou sestavovány jednotlivci, týmy nadšenců a dobrovolníků či komerčními firmami. Distribuce zahrnuje jádro, další systémový a aplikační software, grafické uživatelské rozhraní. Distribuce mají různá zaměření, například výběr obsažených programů, podpora určité počítačové architektury.

Mezi nejznámější distribuce Linuxu patří například Arch Linux, Debian, Fedora, Red Hat Enterprise Linux, Gentoo, SUSE, Ubuntu a mnoho dalších.

Každá distribuce je k dispozici pod různou licencí. Většina licencí má licenci, která nevyžaduje od uživatele žádné poplatky a je většinou možné ji používat i ke komerčním účelům (např. Arch Linux, Debian). Některé distribuce mají placenou jen technickou podporu a některé jsou placené zcela a většinou obsahují i aplikace a nástroje, které pocházejí od firem, které daný SW licencuje (např.: Adobe). V dalším případě je placením zajištěna téměř 100% podpora HW a technické zajištění provozu daného HW s technickou podporou 24/7/365 (např.: Red Hat Enterprise Linux).

Pro vývoj internetových aplikací nebo moderních webových stránek je téměř jedno jaká distribuce se zvolí. Je to jen pouze na tom, s jakou distribucí se uživatel ztotožní a jaká mu nejlépe vyhovuje. Jednou z nevýhod samotného Linuxu je fakt, že uživatel musí být obeznámen alespoň ze základní funkčností linuxové distribuce

a měl by vědět alespoň nutné minimum příkazů pro práci s operačním systémem. Bez patřičných znalostí nebo pomoci někoho jiného je i minimální úprava velmi složitá.

3.2.2 Apache

Apache HTTP Server je softwarový webový server s otevřeným kódem pro všechny majoritní operační systémy a mnoho minoritních. V současné době je nejvíce používaným HTTP serverem.

Vývoj Apache se datuje od roku 1993. Vznikl v NCSA (National Center for Supercomputing Aplications) na Illinoiské univerzitě. První veřejná verze s označením 0.6.2 byla vydána v dubnu 1995. Následovalo kompletní přepsání kódu a založení Apache Group, která je dnes základem vývojářského týmu. Poslední verzí, v době psaní této diplomové práce, je verze s označením 2.4. jenž se vyznačuje hlavně změnou v konfiguračních souborech tak rozsáhlým, že předchozí verze konfigurace bez přepsání nejsou kompatibilní.

Apache podporuje velké množství funkcí. Mnoho funkcí jsou implementovány jako kompilované moduly rozšiřující jádro. Mohou to být funkce podpory programovacích jazyků na straně serveru nebo různé bezpečnostní funkce. Příkladem pod porovaných programovacích jazyků je PHP, Perl a Python. Z bezpečnostních funkcí jsou jistě velmi používané mod_access, mod_auth. Příkladem dalších funkcí je podpora SSL, TLS (mod_ssl), proxy modul(mod_proxy), URL rewriter známý jako rewrite engine z modulu mod_rewrite, konfigurace souborů logu.

Logy z Apache můžou být analyzovány pomocí browseru a skriptů jako AWStat či W3Perl. Což je velmi užitečné pro analýzu chyb nebo nalezení tzv. „bottle necku", případně analýzu používání dané aplikace uživateli. Nastavit se dají i formy chybových zpráv.

Apache podporuje také virtuální hosting. Což je funkce dovolující jedné instanci Apache na jednom fyzickém počítači obsluhovat více webových stránek. Každá stránka může být přístupná z jiné domény. Konfigurace Apache je velmi přehledná a díky změnám v poslední verzi je i velmi modulární, takže se již Apache nemůže

stát nefunkční po aktualizaci. Dříve se totiž veškerá konfigurace uváděla primárně do několika konfiguračních souborů. V poslední verzi jsou veškeré uživatelské úpravy a nastavení uloženy v souborech, které se nepřepisují.

Přestože hlavním cílem Apache není být „nejrychlejším“ webovým serverem, jeho výkon se může srovnávat s HTTP servery, které se orientují na výkon. Místo implementování jedné architektury, Apache poskytuje mnoho tzv. MultiProcessing modulů (MPM) což mu dovoluje přizpůsobit se potřebám systému na kterém běží. Takže výkon je hodně závislý na konkretní konfiguraci MPM uživatele. Apache je navrženo tak, aby latence byla co nejnižší a propustnost co nejvyšší, vzhledem k obsluze více požadavků, tedy zajistit konzistentní a spolehlivé obsloužení požadavků v co nejkratším časovém rámci.

Díky přehledné dokumentaci a velmi rozsáhle komunitě lidí, kteří Apache používají, je velmi jednoduché nasadit tento HTTP server kamkoliv a během několika minut zprovoznit prostředí, ve kterém se bude moderní web nebo webová aplikace nacházet.

3.2.3 MySQL

Michael Widenius, databázový software programátor, byl hlavním architektem jazyka MySQL. MySQL byla z počátku vyvíjena v malé firmě ve Švédsku. Její vznik je datován od roku 1995. MySQL je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Jako jediný kus ze sady LAMP je vlastněn a vyvíjen korporátní společností. Konkretně firmou Oracle. V době prodeje MySQL Oraclu vznikly obavy o zachování komunitní verze. Proto vznikla odnož MySQL, který nese název MariaDB[16] a je vyvíjena komunitou.

MySQL je multiplatformní databáze. Patří do rodiny databází komunikující pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Pro svou multiplatformnost (lze nejen instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky komunitní licenci, má vysoký

podíl na v současné době používaných databázích.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušeních a na úkor funkcí, které ostatní SQL databáze měli. Postupem vývoje byli tyto funkce zařazeny a nyní lze považovat MySQL za plnohodnout SQL databázi. Díky její univerzálnosti lze MySQL použít na jakýkoliv typ dat.

Podstatná je nevýhoda MySQL (taktéž i ostatních SQL databází) v nutnosti správného návrhu databáze. Návrh databáze se řadí k obtížnějším částem vývoje. Správný návrh databáze a dobře vytvořené SQL dotazy dokáží zvýšit výkon celé aplikace nebo webových stránek až mnohonásobně.

3.2.4 PHP

PHP je skriptovací programovací jazyk. Je určený především pro programování dynamických internetových stránek a webových aplikací například ve formátu HTML, XHTML. PHP lze použít i k tvorbě konzolových a desktopových aplikací.

Při použití PHP pro dynamické stránky jsou skripty prováděny na straně serveru. To znamená, že k uživateli je přenášen až výsledek jejich činnosti. Interpret PHP skriptu je možné volat pomocí příkazového řádku, dotazovacích metod HTTP nebo pomocí webových služeb. Syntaxe jazyka vychází z několika programovacích jazyků (Perl, Pascal, C). PHP je nezávislý na platformě, rozdíly v různých operačních systémech se omezují na několik systémově závislých funkcí a skripty lze většinou mezi operačními systémy přenášet bez jakýchkoli úprav.

PHP je velmi modulární skriptovací jazyk. Podporuje mnoho knihoven pro různé účely – mimo jiné i na zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (například: MySQL, ODBC, Oracle, PostgreSQL, MSSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …).

PHP je v poslední době nejrozšířenějším skriptovacím jazykem pro internetové stránky. Oblíbeným se stal hlavně díky jeho jednoduchosti použití bohaté zásobě funkcí.

3.2.5 Perl

Perl je interpretovaný programovací jazyk. Vznikl v roce 1987 a je vytvořený Larry Wallem. S rozvojem internetu se Perl stal velmi populárním nástrojem pro tvorbu CGI skriptů.

Perl zahájil svou éru jako skriptovací jazyk, jenž měl nahradit jazyk AWK. Největšího rozšíření dosáhl ve verzi 4 z roku 1991. Verze 5 přinesla četná vylepšení, především výkonné datové struktury a možnost objektového programování.

Perl umožňuje psát stejné programy různými způsoby. Díky tomuto faktu lze v Perlu napsat na několika řádkách kód, který vykonává totéž jako kód na diametrálně rozdílném počtu řádků. Bohužel čím je program kratší, tak je hůře čitelnější (při zachování stejné funkcionality). Proto Perl získal nezaslouženou pověst jazyka, ve kterém se tvoří nesrozumitelný a neudržovatelný kód.

Silnou stránkou Perlu je velký počet modulů, které usnadňují programátorovi život. Již je přes 18000 volně dostupných modulů třetích stran v Comprehensive Perl Archive Network (CPAN). Díky standardizaci vytváření modulů a přidávání do CPANu je instalace modulů pro použití plně automatizovaná.

Perl podporuje znakovou sadu Unicode, která se stává již řadou let standardem pro znakovou sadu. Modul mod_perl umožňuje web serveru Apache vložení Perlu s výhodami, jako je při vložení PHP. Takže lze velmi jednoduše starší program napsaný v perlu použít s menší úpravou jako webovou aplikaci.

3.2.6 Python

Python navrhl Guido van Rossum v roce 1991. Python je vysokoúrovňový skriptovací programovací jazyk. Python je beztypový interpretovaný jazyk a také objektově orientovaný jazyk. Nabízí významnou podporu k integraci s ostatními jazyky (například s C, C++ a Javou) a přichází s mnoha standardními knihovnami. Sice se jejich počet nevyrovná počtu knihoven jako má Perl, ale i tak je jejich množství velice rozsáhlé. S pomocí knihovny mod_python pro webový server Apache lze taktéž Python využít k psaní webových aplikací nebo webových stránek. Taktéž si rozu-

mí s prakticky všemi trošku více známými databázemi. Umí pracovat stejně dobře s MySQL jako s MongoDB či s jinou SQL a noSQL databází.

Python je velmí silný nástroj, který se v posledních letech těší ze své obliby u programátorů. Zejména začátečníci si pochvalují jednoduchost. Dokonce se tvrdí, že se tento programovací jazyk lze naučit během jednoho týdne. Vůči Perlu a PHP je taky velmi přehledný, takže pozdější úpravy jsou velmi jednoduché. Jako předešlé dva skriptovací jazyky má dynamickou typovost proměnných. Oproti nim má však ale velmi silně deklarovaná datová pole (list, seznam), která při dobré aplikaci zjednoduší kód.

Populárnost Pythonu pomohla vzniknout několika frameworkům, které slouží k vytváření webových aplikací. K nejznámějším patří jistě Django[17]. V něm lze udělat funkční dynamickou webovou stránku během několika málo minut.

3.3 Představení sady MEAN

Druhá, o několik let mladší, softwarová sada MEAN vznikla na základě oblíbenosti několika softwarových balíčků, které časem doby definovaly novodobý standart pro moderní vývoj webu a interaktivních webových aplikací.

První počátky náznaku o možné použitelnosti jednoho jazyka pro vývoj jak serverové části, tak pro vývoj klientské části se datuje k roku 2009. V tomto roce Ryan Dahl využil engine V8 JavaScript od společnosti Google. K tomuto základu přidal několik dalších knihoven a tím vdech život novému softwarovému http serveru pod názvem NodeJS.

MEAN je díky své multiplatformnosti (opravdu lze veškeré části použít na všech populárních platformách a operačních systémech) velmi oblíbený. Sice pořád zůstává problém s počáteční přípravou na daném operačním systému, ale díky integraci do distribučních repozitářů (Linux) a automatickým instalátorům (MS Windows) je počáteční nastavení pro provoz velmi jednoduché a lze vše připravit do několika minut.

Vývoj okolo MEANu je dynamický a poměrně rychlý. Dalo by se dokonce říci,

že jakmile přijde nějaká nová HW novinka, která lze nějak zakomponovat do webových stránek nebo webové aplikace, tak je během několika málo dní zakomponována také právě do MEANu. Což je velmi lukrativní krok pro technologické nadšence, ale i pro pohon trhu právě těchto HW novinek.

MEAN má velmi silnou komunitu. Na internetu lze nalézt velké množství tutoriálů, videonávodů a to nejen od nadšenců, ale od velkých firem (Microsoft, Google). Velké firmy totiž apelují na integraci vývoje projektů do jejich cloudových prostředí (např. Microsoft - Azure).

3.3.1 MongoDB

MongoDB je multiplatformní dokumentová databáze. Je jednou z NoSQL databází. Místo relačních databází využívajících tabulky používá dokumenty formátu JSON a dynamické databázové schéma, které umožňuje vytváření a integraci dat pro aplikace jednodušeji a rychleji.

MongoDB byla vyvinuta v listopadu roku 2007 softwarovou společností 10gen. V roce 2009 se z projektu stal opensource. Od roku 2009 je MongoDB implementováno jako backend řešení množství velikých stránek a služeb včetně stránek jako je eBay, a SourceForge. MongoDB je nejpopulárnější NoSQL databázový systém.

Díky své všestrannosti a jednoduchosti je MongoDB podporována v prakticky ve všech více používanějších skriptovacích a programovacích jazycích. Ve spojitosti s NodeJS je několik implementací jak právě s MongoDB pracovat a jak přistupovat k datům. Vezmeme-li v potaz, že ve webových aplikací se převážně přeposílají data ve formátu JSON. Takže by se dalo říci, že i díky tomuto faktu je předávání dat mezi databází a aplikací rychlejší.

Mezi nejvíce používané moduly pro přístup k MongoDB z NodeJS je dozajista modul s názvem Mongoose[18]. Modul umožňuje na několika řádcích implementovat přístup a datovou strukturu modelu jak pro zápis tak, tak pro čtení.Což ocení zejména programátoři, kteří nechtějí nebo neví jak samotná databáze funguje, ale přesto chtějí mít plnou kontrolu nad daty. Kontrolu nad daty taktéž zajišťuje možnost validace dat jak při zápisu, tak při čtení.

3.3.2 Express

Express je jedním z mnoha modulů pro NodeJS. Byl vytvořen jako minimalistický framework pro vytváření webových aplikací. Použitím Expressu lze lehce do aplikace přidat Middleware, který by se jinak implementoval podstatně hůř. Nejsilnější výhodou tohoto frameworku můžeme považovat možnost napsat si vlastní callback funkci požadavek na server. To v praxi znamená, že lze efektivně napsat SPA nebo RESTfull aplikaci. Taktéž lze pomocí dalších modulů doplnit další funkcionality (například ověřování uživatele), tím se stává právě Express velice užitečným nástrojem pro vznik moderní webové aplikace.

Express patří mezi frameworky, které si daly za cíl optimalizovat samotné zpracovávání požadavků na server. Výsledky v poslední verzi (4.2) jsou působivé. Testy s výsledky v porovnání Expressu, NodeJS a Apache lze nalézt následující kapitole.

Další velmi povedenou vychytávkou se může jevit implementovaný debug mód, ve kterém lze v reálném čase vidět veškeré dotazy na server a jejich zpracování i s chybami. Sice se nedoporučuje testovat tímto způsobem celou webovou aplikaci, ale pokud chce programátor sledovat reálný provoz, pak je tato funkce užitečná.

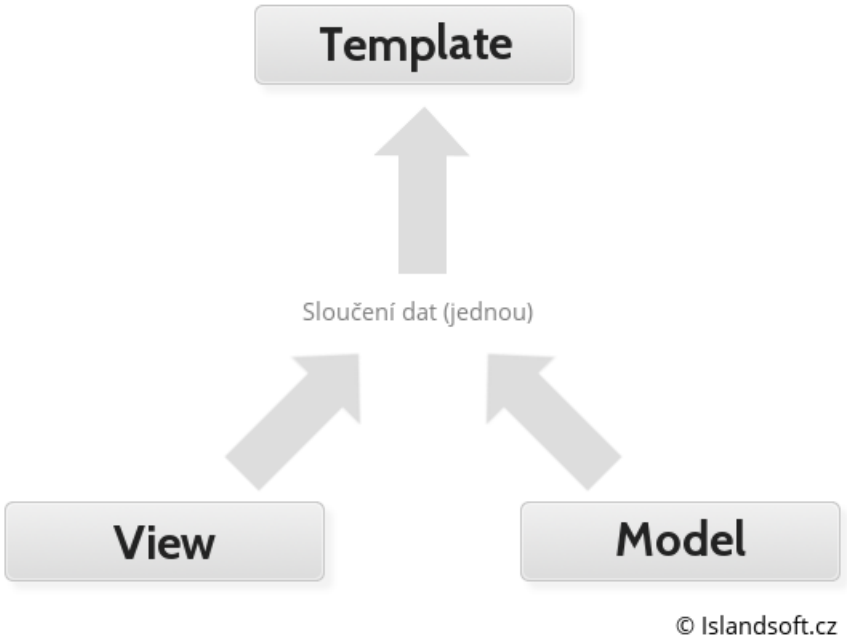
3.3.3 AngularJS

Nástupem technologií HTML5 se obliba JavaScriptu rozrostla. Aplikace napsané pomocí JavaScriptu se stávají čím dál víc komplexní a náročnější na udržení čitelnosti kódu. Dosud nejznámější knihovna jQuery[19] na ideální vývoj přestává stačit. Knihovna má totiž jednu velkou nevýhodu a to, že míchá dohromady aplikační logiku, zpracování událostí a manipulaci s DOM. Což u menších aplikací a webových stránek není takový problém. Větší aplikace se stávají velmi nepřehledné a je velmi složité a obtížné udržet kód čitelný tak, aby předělání kódu nebo přidání nové funkce nepředcházelo studii kódu původního. V takovém případě je ideální použít architekturu MVC nebo nějaký její derivát.

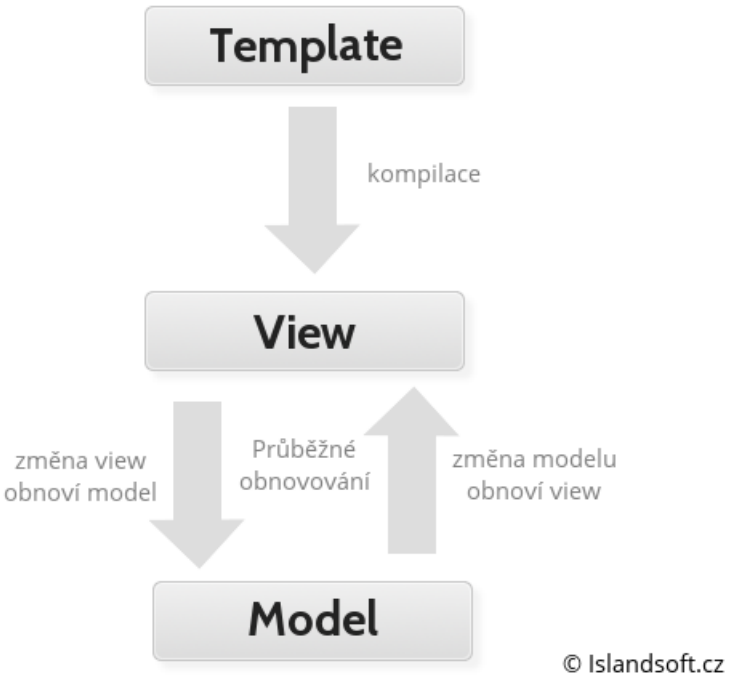
V roce 2009 se firma Google rozhodla vyvinout právě pro tyto účely vlastní javascriptový framework. Pojmenovala ho AngularJS a je primárně zaměřený na

single-page aplikace (SPA) -typ webové aplikace, která většinou používá server jen jako zdroj a úložiště dat. Data jsou potom kompletně vykreslována JavaScriptem na straně klienta. Krom již zmíněné architektury se honosí AngularJS dalšími zajímavými vlastnostmi. Díky nimž se stal jedničkou ve vývoji moderních webových aplikací mezi MVC javascriptovými frameworky.

Specifickou vlastností právě AngularJS je hlídání změn v hodnotách proměnných. K tomuto účelu slouží vlastní specifická mezivrstva, která je označována jako $scope. Právě $scope slouží k tomu, aby promítl změny v controlleru nebo ve viewu v závislosti na změnu obsahu dat. Této vlastnosti se říká "two-way data binding". Pro lepší představu konceptu bindingu se lze podívat na obrázky níže. Na nich lze vidět princip "one-way data bindingu" (obrázek 3.1) a "two-way data bindingu" (obrázek 3.2). Z obrázků lze vidět jejich odlišnost. Nutno podotknout, že většina MVC frameworků používá "one-way data binding".



Obrázek 3.1: Princip "one-way data bindingu"



Obrázek 3.2: Princip ”two-way data bindingu”

Další výhodou AngularJS je použití návrhového vzoru Dependenci injection. Jedná se o návrhový vzor, který řeší závislosti mezi jednotlivými komponentami aplikace. AngularJS obsahuje zabudovaný subsystém, který řeší dependenci injection napříč celou aplikací. Umožňuje přesně specifikovat, které jiné komponenty jsou použité uvnitř konkrétní komponenty.

3.3.4 NodeJS

Jak již bylo uvedeno v úvodu, tak NodeJS je softwarový http server založený na enginu JavaScript od společnosti Google. NodeJS dokáže zastoupit kombinaci Apache a PHP (případně jiný jazyk pro psaní dynamického kódu pro web a jiný http server).

NodeJS se vůči výše uvedené kombinaci odlišuje svými vlastnostmi. Tím lze NodeJS lépe použít na aplikace, které dané vlasnosti využije. Mezi vlastnosti bych rád uvedl rozdílné běhové prostředí. Pokud přijde požadavek na web realizovaný v PHP a webovém serveru Apache, pak se vytvoří nové vlákno. Vlákno obslouží požadavek a spolu s tím se vytvoří také prostředí pro běh PHP aplikace. Potom se nahrají systémové proměnné a přeloží se PHP kód, který se spustí se vstupními

daty. V NodeJS je tento proces napsaný přímo v sobě. Žádosti jsou tedy přidávány ve formě "událostí" do seznamu událostí určených ke zpracování. Server pak projíždí seznamem událostí a postupně je zpracovává. Z toho lze vypozorovat, že nedochází ke zvýšené režii při příchodu více požadavků. Další nespornou výhodou je také to, že se při zpracování požadavku neskončí celý proces, ale běží dál a čeká na další události pro zpracování. V aplikaci tak můžeme ukládat průběžné informace, které bychom museli při dalším požadavku pracně počítat znova. Takže lze pomocí této vlastnosti vyřešit základní formu cachování.

Další zajímavým faktem o NodeJS je používání callback funkcí. Což znamená, že vývoj klientské části a serverové části v JavaScriptu se moc neliší. Zajisté tento fakt zlepšuje přehlednost i strukturu celé aplikace.

Při vývoji NodeJS bylo schválně zvoleno jednovláknové zpracování. Při vícevláknovém zpracovávání událostí by totiž docházelo k problémům se synchronizací, zamykáním atd. Proto bylo zvoleno pouze jedno vlákno byla použita event-driven architektura. Tato architektura umožňuje při načtení dat vytvoření události. Při události se zavolá nastavená callback funkce printResult. Než přijde výsledek z čtení souboru může NodeJS pokračovat ve zpracovávání dalších událostí. Takto může NodeJS zpracovávat I/O operace asynchronně. Pomocí asynchronního modelu se tak může vyřídit více I/O operací najednou a tím snížit celkový čas pro zpracování celého požadavku. Naproti tomu v PHP se nejprve čeká na načtení uživatelských dat z databáze a až poté se pošle požadavek na načtení následujících dat.

NodeJS má k dispozici několik desítek tisíc modulů. Moduly lze instalovat pomocí integrovaného nástroje npm[20], který poskytuje jednotné místo pro uchovávání modulů. Díky npm lze instalovat, spravovat a vytvářet moduly. Pakliže si uvědomíme, že i nástroj npm je multiplatformní, pak je každý projekt lehce přenositelný na jiný systém/hardware s možností doinstalování závislostí právě s pomocí npm.

3.4 Srovnání sady LAMP a sady MEAN

V předchozích kapitolách jsme se seznámili se sadou LAMP a se sadou MEAN. Z popisu lze usoudit jakým směrem se každá sada vydává. Jakou sadu tedy vybrat pro vytvoření nového projektu? Když opomeneme osobní preference programátora. Kde je výběr sady již rozhodnut. Pak zbývá jen několik možností. Pokud by výběr spočíval na trendu ve webových technologií, pak by MEAN byl jasným vítězem. Hloubavější lidi by však mohli namítat, že trend není vše. Pak tedy vyplyne otázka jak a co porovnávat? První odpověď by nejspíše byla: Je opravdu NodeJS rychlejší ve zpracování uživatelských požadavků než Apache + PHP? Další by jistě jistě následovala otázka je lepší SQL nebo noSQL databáze? Jistě by se našlo mnoho dalších otázek, ale tyto dvě jsou nejožehavějšími tématy diskuse mezi těmito sadami.

3.4.1 NodeJS vs Apache + PHP

V prvním testu se zaměříme na test rychlosti a dostupnosti http serveru s dynamicky generovaným jednoduchými webovými stránkami. Nutno předem říci, že pro tento test byl použit Apache + PHP jako nejvíce zastoupenými prvky v množině dostupných alternativ. Tímto rozhodnutím je tedy výsledek trochu ovlivněn ve prospěch NodeJS, ale cílem je demonstrovat jaký je rozdíl pro normálního uživatele, který si svojí sadu nainstaluje podle návodů na internetu.

Testy byly prováděny na virtuálním stroji s těmito parametry:

- CPU: 2Ghz
- RAM: 4GB
- HDD: 20GB
- OS: Ubuntu 10.04 Lucid 64 bit

Testy byly prováděné pomocí utility ApacheBench[20] a pomocí dstat[21]. Byly provedeny 2 testy v několika opakování. Grafy a výsledky jsou průměrem výsledků.

Zdrojový kód pro NodeJS:

```
var sys = require('sys'),
http = require('http');

http.createServer(function(req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('<p>Ahoj světe!</p>');
  res.end();
}).listen(8080);
```

Zdrojový kód pro PHP:

```
<?php echo '<p>Ahoj světe!</p>'; ?>
```

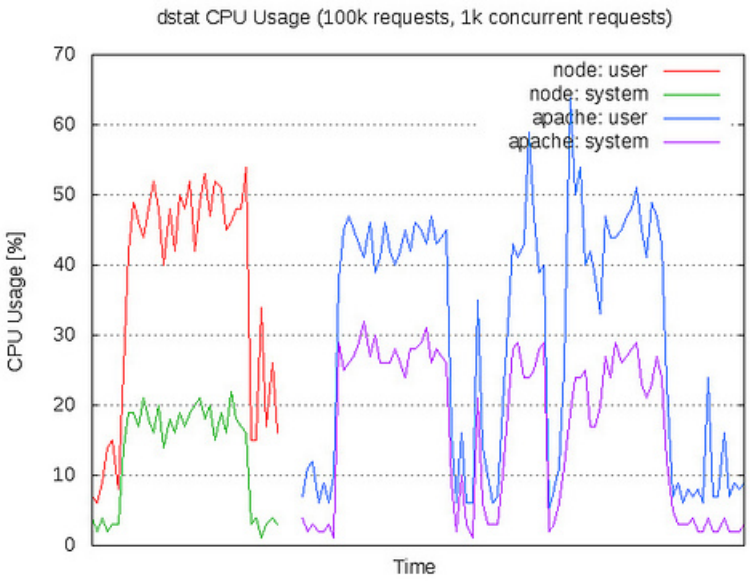
Výsledné statistiky a grafy pro 1. test se 100 000 dotazy s možností obsluhy 1000 dotazů za sekundu:

- NodeJS:

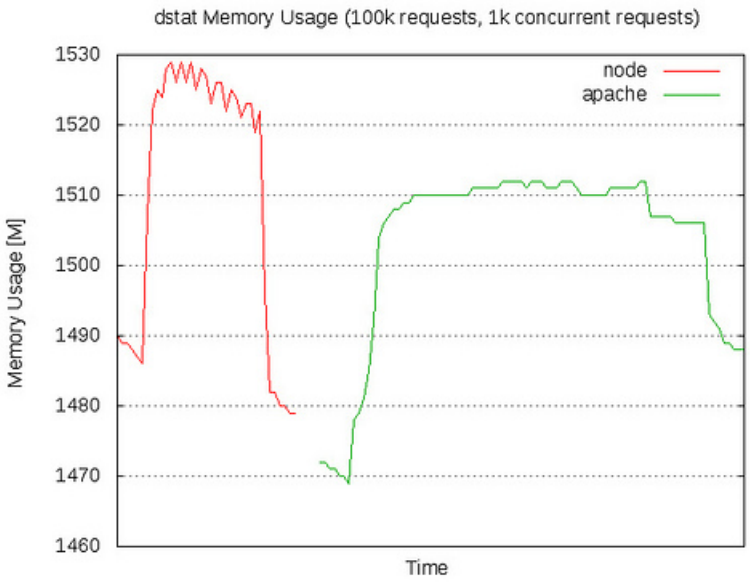
```
Concurrency Level:      1000
Time taken for tests:   21.162 seconds
Complete requests:      100000
Failed requests:        147
```

- Apache:

```
Concurrency Level:      1000
Time taken for tests:   121.451 seconds
Complete requests:      100000
Failed requests:        879
```



Obrázek 3.3: Graf vytíženosti CPU při 1. testu



Obrázek 3.4: Graf vytíženosti RAM při 1. testu

Výsledné statistiky a grafy pro 1. test se 1 000 000 dotazy s možností obsluhy 20 000 dotazů za sekundu:

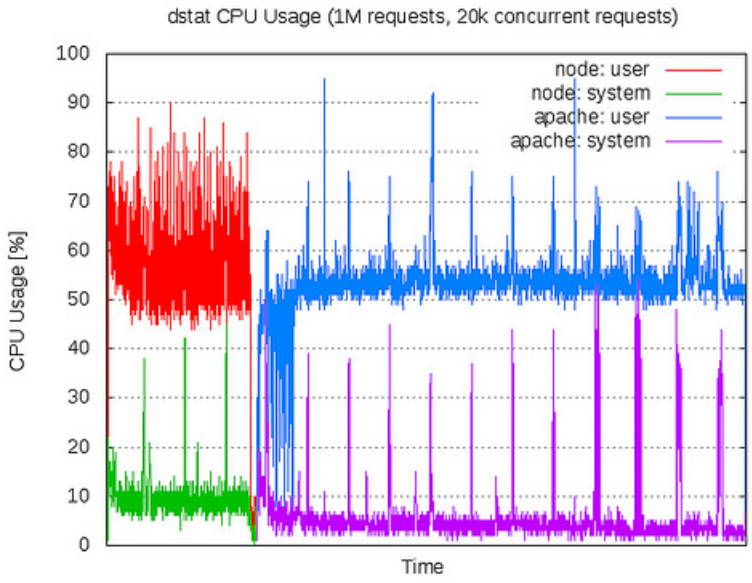
- NodeJS:

```
Concurrency Level:      20000
```

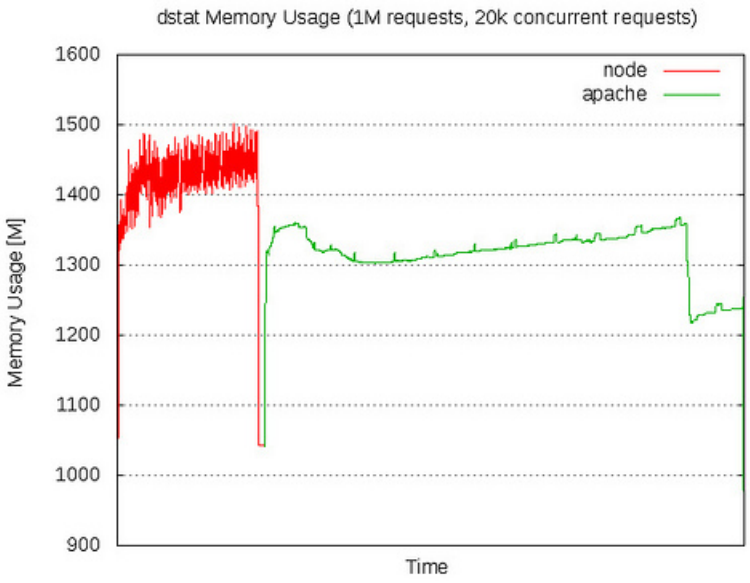
```
Time taken for tests:   1043.076 seconds
Complete requests:      1000000
Failed requests:        25227
```

- Apache:

```
Concurrency Level:      20000
Time taken for tests:   3570.753 seconds
Complete requests:      1000000
Failed requests:        2617614
```



Obrázek 3.5: Graf vytíženosti CPU při 2. testu



Obrázek 3.6: Graf vytíženosti RAM při 2. testu

Z testů lze vyvodit několik informací. NodeJS je o mnoho rychlejší v základním nastavení než Apache + PHP. Lze taky vidět, že Apache zamítne při vyšší zátěži o mnoho více požadavků než NodeJS. Na druhou stranu Apache si zabere pro zpracování méně systémových požadavků. V poměru výkon/prostředky však vyhrává NodeJS.

3.4.2 SQL vs NoSQL databáze

Mnoho lidí se pře o to zda je lepší používat SQL nebo NoSQL databáze. Odpověď však není jednoznačná. Každý vývojář by při návrhu aplikace měl věnovat velký kus času při výběru vhodné databáze. Jelikož oba dva typy mají své klady a zápory. Pakliže se omezíme na vývoj moderních webových aplikací a webových stránek pak můžeme vybírat dle těchto aspektů:

- Struktura databáze - normalizace dat
- uložení struktury dat

Při vývoji aplikace se mnohdy stane, že prvotní návrh databáze je mylný, špatný nebo lehce opomíjí skutečnou potřebu pro uchovávání dat. A protože NoSQL databáze ukládají tzv. kolekce (kolekce je struktura objektu obsahující daná data),

pak lze už z definice předpokládat, že úprava struktury je o mnoho jednodušší než přepsání celé normované SQL databáze. Tím se vývoj aplikace radikálně zrychlí a programátoři mohou svůj čas věnovat jiným věcem. Na druhou stranu lze namítnout, že NoSQL databáze nemohou provést složitější dotazy a úprava dat ve velké databázi trvá o dost větší dobu než v SQL databázi. Tyto argumenty jsou pravdivé. Proti těmto argumentům však stojí fakta, že málokdy je opravdu využívána velká databáze a že mnohdy je právě dotaz směřovaný na sestavení nějakého objektu.

Dalším aspektem se jistě jeví způsob uchovávání dat. Je lepší uchovávat celé objekty v kolekci nebo vše postupně sestavit pomocí SQL dotazů? V trendu dnešní doby je jistě prospěšnější využít již hotové kolekce, které jsou distribuovány v nějakém formátu (např.: JSON, XML).Pokud aplikace dostane od serveru takto získanou kolekci lze celou kolekci předat další aplikaci ihned bez žádných mezikroků. Toto řešení velmi zrychluje komunikaci a zamezuje vzniku chyb při převodu dat do jiného formátu nebo jeho úpravě.

V rámci porovnání by se dalo říci, že v dnešní době je výhodnější používat NoSQL databáze. Hlavně při vývoji, kde změny jdou o dost jednodušeji než v SQL databázích. NoSQL databáze jsou taktéž rychlejší pro čtení a zápis i když neposkytují v dotazování takových možností jako má SQL. Jasnou výhodou NoSQL databází je jednoznačně uchovávání dat v kolekcích, které dost často jsou ve formátu, který je standartem v předávání dat v moderních webových aplikací.

3.5 Závěr srovnání a výběr sady

Po zodpovězení dvou hlavních otázek při porovnání sad by se dalo usoudit že vítězem se stala sada MEAN. Mezi nespornou výhodu sady MEAN je možnost psání veškerého kódu pomocí jednoho programovacího jazyka. Což v případě použití ať už LAMPu nebo jiné variace sady není reálné. Podpora velkých firem MEANu též prospívá a její propagace právě těchto firem přispívá v oblibě u programátorů. Její vývoj je již mnohonásobně rychlejší než u LAMPu což s novými možnostmi hardwaru je neocenitelný argument proč si vybrat tuto sadu. V praktické části bude

pro projekt školního interaktivního knihovního systému použita právě sada MEAN. Nejen z důvodů výše uvedených, ale i z možnosti snadné rozšiřitelnosti projektu, která vyplývá z principu, návrhu a funkcionality MEANu.

4 Praktická část

V následujících kapitolách se zaměříme na vývoj interaktivního knihovního systému. Systém bude vytvořen jako webová aplikace. Jádrem webové aplikace bude sada MEAN, která byla popsána v teoretické části. Cílem knihovního systému je ulehčit a zpřehlednit uchovávání knižních a odborných publikací na katedrách. Při návrhu aplikace byl použit obecnější návrh, a to z důvodu možnosti změny zaměření aplikace za minimální čas.

4.1 Požadavky na knihovní systém

Před samotným vývojem knihovního systému bylo nutné si stanovit vlastnosti aplikace. Jak již v úvodu bylo řečeno, tak stále zůstává v knihovnách nutnost zapsání o výpůjčce u personálu. Tato interaktivní webová aplikace si dala za cíl tento krok téměř eliminovat tak, aby si publikaci mohl vypůjčit kdokoliv. Prvotní nutností bylo vyřešit identifikaci publikací tak, aby byla jednoznačná možnost určení konkretní publikace a to i za předpokladu identifikace kopií publikace. Částečně se mohlo inspirovat v systému, které se používají dodnes. Tedy použitím bar kódu[22]. Pro tyto účely byl vybrán speciální typ bar kódu. Konkrétněji QR kód[23], který se stává v posledních letech velmi populárním a lze jej najít téměř na každém obalu nebo na plakátu či časopise. Jeho nespornou výhodou oproti klasickému EANu[24] je možnost uchovávání více informací, což se velmi hodí. Toto rozhodnutí velmi dopomohlo k vytvoření dalších požadavků na aplikaci:

- Aplikace typu server-klient
- Použití REST API

- Použití QR kódu na identifikaci publikace
- Možnost vypůjčení publikace pomocí webkamery
- Nutnost autorizace uživatele a vytvoření uživatelských oprávnění
- Možnost přidání, úprava či smazaní publikace do/z databáze
- Modulárnost aplikace s možností jednoduchého rozšíření
- Vytvoření aplikace pro mobilní telefony

Výše uvedený seznam uvádí požadavky, které vycházejí z principu fungování knihovních systémů a z požadavků budoucích uživatelů výsledné aplikace.

4.2 Instalace a příprava serveru

Pro vývoj a první instanci interaktivní knihovní aplikace byl zvolen operační systém Linux. Linux byl zvolen nejen z hlediska nižších HW režií, ale i proto, že v daném místě se využívají virtuální servery postavené na Linuxu. S tím souvisí i výběr distribuce. Byla použita známá distribuce, která je určena pro serverové nasazení. Její jméno je CentOS[25] a byla zvolena v 64-bitové architektuře. Po instalaci minimální serverové verze (bez grafického rozhraní) je nejprve nutné provést update systému a posléze upgrade:

```
su -c 'yum update'
```

Posléze je taktéž dobré nastavit pravidelné aktualizace. Což je vhodné kvůli bezpečnostnímu riziku:

```
su -c '/sbin/chkconfig --level 345 yum on; /sbin/service yum start'
```

Po aktualizaci systému je potřeba doinstalovat balíčky, které jsou potřeba pro sadu MEAN a její závislosti. Pro nainstalování MongoDB databáze je nutné provést tyto operace:

- Přidání repozitáře MongoDB

```
# vytvoříme soubor
$ touch /etc/yum.repos.d/mongodb-org-3.0.repo

# vložíme do něj tyto data
[mongodb-org-3.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/
$releasever/mongodb-org/3.0/x86_64/
gpgcheck=0
enabled=1

# pro instalaci nejprve aktualizujeme repozitáře
# a potom spustíme instalaci
$ sudo yum update && sudo yum install -y mongodb-org

# posléze je nutné povolit v SELINUXu požadovaný port
# pro naslouchání
$ sudo semanage port -a -t mongod_port_t -p tcp 27017

# nastavení cesty a práv pro databázi
$ sudo mkdir -p /data/db && sudo chown -R root /data/db

# spustění služby
$ sudo service mongod start

# nastavení spuštění služby po startu systému
$ sudo chkconfig mongod on
```

- NodeJS

```
# stažení a rozbalení zdrojového kódu
```

```
$ cd /usr/src
$ wget http://nodejs.org/dist/v0.10.32/node-v0.10.32.tar.gz
$ tar -zxf node-v0.10.32.tar.gz
$ cd node-v0.10.32
$ ./configure && make && sudo make install

# povolení NodeJS v SELINUXu
$ chcon -t execmem_exec_t node
```

Po instalaci všech těchto balíčků je server schopný obsluhovat vytvořenou aplikaci a lze na něm i dále aplikaci dále rozvíjet. Pakliže je na serveru puštěný firewall, potom se musí ještě povolit port, na které naslouchá NodeJS.

4.3 Serverová část knihovního systému

Serverová část knihovního systémů se zaobírá zpracováním http dotazů ze strany klienta. Je zodpovědná za uchovávání dat a jejich úpravy v reálném čase. Hlavní komponenta serverové části je NodeJS, který obstarává komunikaci s MongoDB databází a kontroluje, zda dotazy, které přicházejí na server přicházejí i s patřičným oprávněním. Oprávnění kontroluje Middleware, který je implementován jako modul pro NodeJS. Pro serverovou část je nutné nainstalovat pomocí npm následující moduly:

- express
- express-session
- cookie-parser
- body-parser
- node-restful
- mongoose

- bcrypt
- passport
- passport-local

Veškeré moduly je nutné instalovat pomocí příkazu:

```
$ sudo npm install #NAZEV_MODULU --save
```

parametr save zajišťuje možnost zapsání informace o instalaci do hlavního informačního souboru. Při nutnosti zprovoznit serverovou část na jiném stroji nebo virtuálním stroji stačí potom přenést všechny soubory a pomocí příkazu

```
$ npm install
```

Což zapříčiní instalaci všech modulů v patřičných verzích, které se nacházejí v hlavním informačním souboru. Což je velmi praktické na replikaci celé webové aplikace.

První pět modulů slouží k instalaci frameworku express a pomocných knihoven pro parsování a úpravu dat. Mongoose modul slouží ke komunikaci s MongoDB databází a lze pomocí tohoto modulu nadefinovat velmi lehce schémata kolekcí. Poslední tři moduly slouží jako Middleware pro ověřování uživatelských práv a následné přesměrování či předání dat ze serveru klientské části aplikace.

4.3.1 Návrh souborové struktury

Pro lepší orientaci ve zdrojových kódech a i pro přehlednost je nutné zvolit dobře souborovou strukturu serverové části aplikace. Jako vodítko může posloužit architektura MVC, která logické rozdělení souborů podporuje. Předem je nutno si určit, zda je vhodné mít v této struktuře místo i na složku s databází. Dle autora je však ideálním místem pro databázi takové umístění v systému, která mu určí sama distribuce. Tím se předchází možným komplikacím, které můžou nastat buď v komunikaci nebo s přístupem v rámci SELinuxu. Po rozvaze vypadá souborová struktura aplikace takto:

```
..
controllers
|--> controller1.js
|--> controller2.js
models
|--> model1.js
|--> model2.js
node_modules
|--> modul1.js
|--> modul2.js
index.js
package.json
passport-init.js
```

Z výše uvedeného schématu lze vidět, že struktura je sice velmi jednoduchá, ale je i velmi efektivní pro orientaci. Ve složce controllers jsou po souborech jednotlivé controllery, které zajišťují kompletní logiku v daných částech programu. Ve složce models jsou obsaženy soubory obsahující model pro každou kolekci. Tento model je použit knihovnou Mongoose ke komunikaci s MongoDB. Složka Node_modules je generována samotným npm a slouží k uchovávání nainstalovaných modulů, které jsou využívány. Soubor index.js je hlavním souborem serverové části a slouží k inicializaci serveru a popsání logiky komunikace a ověřování uživatelských práv. Package.json je soubor, ve kterém se uvadí informace o projektu a následný seznam instalovaných modulů. Díky tomuto souboru lze celý projekt přenést a zprovoznit díky jednomu příkazu. Soubor passport-init.js slouží k inicializaci modulu passport, který slouží právě ke komunikaci s Mongoose a v něm se určuje jakým způsobem a jakou logikou se ověření uživatele provádí.

4.3.2 Návrh oprávnění uživatelů

Při návrhu oprávnění uživatelů je vhodné si předem uvědomit kolik úrovní oprávnění musí požadovaná aplikace mít. Případně jaké operace jaká úroveň oprávnění může

provádět. Typicky můžeme použít základní ideu:

- Nepřihlášený uživatel
- Přihlášený uživatel
- Knihovník
- Správce

V tomto schématu úrovní oprávnění můžeme již zakomponovat možné akce. Nepřihlášený uživatel nemá žádné možné akce. Nemůže ani prohlížet databázi publikací, která se v knihovním systému nachází. Toto rozhodnutí vyplývá z faktu, že uživatel, který nemá práva, tak nejspíše ani nestuduje na TUL či nijak není vůči TUL vázán.

Přihlášený uživatel má právo prohlížet, vyhledávat, půjčovat a vracet publikace. Tyto akce jsou umožněny na základě ověření u knihovníka, který daného uživatele registruje. Potom už registrovaný uživatel nemusí za knihovníkem vůbec chodit a může si publikace libovolně půjčovat a vracet.

Knihovník je člověk, kterého do systému přiřadil správce. Knihovník má práva registrovaného uživatele. K těmto právům získává ještě práva na vytvoření a úpravu uživatele a publikací. Knihovníků může být libovolné množství, ale žádný knihovník nemůže nastavit práva dalšímu knihovníkovi.

Správce je nejvyšší úroveň oprávnění. Je pouze jediný a jeho účet je vytvořen a nastaven při prvotní inicializaci aplikace. Tato úroveň má veškeré možnosti jako předešlé úrovně a má ještě k tomu možnost přidávání knihovníků. Tato úroveň je připravena na další možný rozvoj aplikace.

4.3.3 Návrh struktury databázových kolekcí

MongoDB ukládá do své databáze objekty, které se zapouzdřují do tzv. kolekcí. Kolekce slouží jako předpis popisu uchovávaného objektu. Každý objekt v MongoDB databázi je uchován a lze ho rozlišovat jen pomocí unikátního klíče. Proto je vhodné, aby si strukturu objektu, tedy kolekci, uchovala i aplikace. K tomuto účelu slouží právě modul Mongoose a modely kolekcí, které jsou uloženy podle předpisu.

Níže uvádím několik hlavních kolekcí, které jsou popsány funkcionálně. Ve zdrojovém kódu modelu jsou definovány dle specifikace Mongoose a MongoDB:

```
# model Fakulta

fakulta
|--> název
|--> zkratka # unikatni identifikator
|--> kontakt

# model typ publikace

typ_publikace (číselník typů publikací)
|--> název
|--> typ # 0 --> kniha | 1 --> Vědecká práce
   # 2 --> Bakalářská práce | 3 --> Diplomová práce
   # 4 --> jiná

# model publikace
|--> zkratka_fakulty
|--> autor
|--> oponent
|--> slova # klíčová slova
|--> název
|--> anotace
|--> vedoucí
|--> konzultant
|--> umístění

# model vypůjčení
|--> id_publikace
```

```
|--> id_osoby
|--> datum

# model osoby
|--> fakulta
|--> oprávnění
|--> titul
|--> jméno
|--> příjmení
|--> login # nejčastěji login ze STAGu
|--> heslo
|--> telefon
|--> email
```

Z modelů lze vypozorovat vzájemnou závislost. Tato závislost však není dána MongoDB databází, ale logikou aplikace. Každý záznam v databázi je definován unikátním klíčem. Defaultně je jeho název ID, ale jak je z modelu patrné, tak některé názvy tohoto klíče byly přepsány (například model fakulty, kde by bylo unikátní jak ID, tak zkratka fakulty). Všechny výše uvedené modely jsou plně dostačující na provoz fakultního knihovního systému za předpokladu vypůjčení publikace na libovolně dlouhou dobu.

4.3.4 Návrh REST API

REST implementuje čtyři základní metody, které jsou známé pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu. Aby bylo možné tyto metody používat, je napřed nutné v serverové části aplikace vytvořit návaznost na modely a vytvořit callback funkce, které ošetří samotné zpracování dotazu. Což se v NodeJS ošetřuje ve směrování dotazů. Ve zdrojovém kódu se definice pro všechny kolekce moc nelíší, proto je níže uvedena jen deklarace jedné kolekce. Ostatní se liší jen názvem a danou cestou v API.

V této části implementace je rovněž vhodné použít Middleware pro ověření přihlášení uživatele a jeho úroveň oprávnění. Výsledná ukázka z kódu, kde je vidět jak Middleware, tak obsluha dotazu:

```
router.get('/publication', function (req, res) {
 if (!req.user || req.user.status !== 'ENABLED' || req.user.level < 1) {
   return res.redirect('/login');
  }

  res.render('publication', {title: 'Publikace', user: req.user});
});
```

Z ukázky výše lze vidět obsluhu dotazu na seznam všech publikací. Pro prohlížení seznamu je nutné mít minimální práva úrovně 1. Což je konstanta pro registrovaného uživatele. Pokud je tato podmínka splněna, pak je vrácen seznam publikací, pokud tak není, pak vrácen na stránku s přihlášením.

4.3.5 Závěr serverové části knihovního systému

Z výše uvedeného textu byla vytvořena serverová část interaktivního knihovního systému. Serverová část je realizována za pomocí NodeJS, MongoDB a potřebných modulů. Aplikace je napsána tak, aby dokázala obloužit klienta i bez nutnosti použití GUI, za co nejnižší režie. Serverová část je napsána tak, aby budoucí rozšiřování či pozměňování bylo co nejrychlejší. Taktéž byla vyřešena přenositelnost serverové části pomocí informací o závislostech v souboru package.json.

4.4 Klientská část knihovního systému

Klientská část knihovního systému je napsána v jazyce JavaScript za pomocí frameworku AngularJS. Tato část byla navržena pro potřeby uživatele tak, aby jednoduše našel veškeré informace o publikaci a měl možnost si publikaci jednoduchým způsobem vypůjčit nebo vrátit. Jedním ze způsobů manipulace s publikacemi byl

vybrán QR kód, který v sobě obsahuje veškeré nutné informace o publikaci. V dnešní době má webkameru nebo kameru v telefonu skoro každý. Jelikož i tak se může vyskytnout situace, kdy nelze kód načíst, tak stále je nutno počítat s možností ručního zadání id publikace. Tyto dva způsoby umožňují manipulaci bez pomoci další osoby.

4.4.1 Návrh adresářové struktury

Budeme-li vycházet z rozvahy statických webů dřívější generace, kde každý typ souboru měl svůj vlastní adresář. Dále pak vezmeme v potaz potřebu rozdělení dle MVC architektury AngularJS a oboje logicky spojíme, pak nám vychází nejspíše téměř ideální adresářová struktura pro klientskou část aplikace. Po rozvaze vypadá souborová struktura aplikace takto:

```
..
images
|-->obr1.jpg
|-->obr2,jpg
scripts
|-->controllers
|---->controller1.js
|---->controller2.js
|-->app.js
styles
|-->style.css
|-->mobile.css
views
|-->main.html
|-->publication-add.html
index.html
404.html
```

Z názorné ukázky struktury dat je patrné, že statická data jsou oddělená od dynamických. Taktéž platí, že je oddělena logika od obsahu. Jádrem klientské části aplikace je jistě adresář scripts. V tomto adresáři se nachází veškerá logika aplikace. Díky vlastnosti AngularJS je potom velmi snadné odkazovat na statické pohledy v adresáři views a data v nich opravovat pomocí "two-way data bindingu".

4.4.2 Souhrn funkcí a jejich implementace

Jak již bylo v úvodu řečeno, tak klientská část aplikace by měla zejména zaujímat místo knihovníka. Pakliže si představíme co vše knihovník dělá, tak se nám jistě vybaví tyto funkce:

- Procházení seznamem publikací
- Hledání určité publikace na základě kritérií
- Půjčování a vracení publikací V případě knihovníka lze doplnit klientskou část ještě o funkce:
- Přidání/úprava/vymazání publikace
- Přidání/úprava/vymazání registrovaných uživatelů

Když si pečlivě projdeme strukturu modelů kolekcí pro MongoDB v serverové části lze vidět, že díky návrhu lze většinu funkcí provádět triviálním využitím RESTu. AngularJS se pouze postará o vykreslení a o vypsání dat do pohledu.

Zajímavými funkcemi jsou jistě půjčování či vrácení publikace a jejich založení. Na tyto funkce je ovšem potřeba napsat dodatečné moduly pro AngularJS tak, aby šlo využívat QR kódy ve spolupráci s webkamerou nebo fotoaparátem v telefonu.

Začneme nejprve s vypůjčováním a vracením publikací. Od HTML verze 5 je možné používat videozdroje jako součást webových stránek nebo aplikace. S touto znalostí lze již použít knihovnu, která zpracovává QR kódy. Informace z QR kódu lze opět použít jako parametr pro REST a tím je vše vyřešeno.Implementace vytvořeného modulu je potom do AngularJS velmi jednoduchá a vypadá následovně:

```
#view publication_back.html
...
<qr-scan ng-success="onOK(data)" ng-error="onKO(error)") />
...

#controller pro publication_back
App.controller('qrctrl',['$scope',function($scope)]){
$scope.onOK = function(data){
...
console.log(data)
...
}
$scope.onKO = function(error){
...
console.log(error)
...
}
}
```

V této ukázce z kódu lze jasně pochopit, že při správném přečtení dat jsou data zpracována a server po autorizaci uživatele odpoví a daná publikace se ukáže na straně klienta. Uživatel pak jen zvolí akci, kterou chce s publikací provést.

Podíváme-li se na problematiku přidání publikace knihovníkem a vynecháme vyplnění formuláře s informacemi, tak nám zůstane jen nutnost vygenerování QR kódu. Implementací QR kódu pro JavaScript existuje nepřeberné množství. Po vybrání patřičné knihovny bylo nutné jen opět zapouzdřit knihovnu do modulu pro AngularJS. Data, ze kterých je generován QR kód jsou získány ze zpětného testu zápisu do databáze na serveru. Potom vykreslení QR kódu v naší aplikaci vypadá následovně:

```
# view publication-add.html
...
```

```
<qr ver="{{version}}" error-level="{{level}} data={{publication.id}}">
</qr>
...
```

Daný QR kód se ukáže na obrazovce ve formě obrázku a kód lze již jen vytisknout a nalepit na publikaci. Aby byla možnost ručního zadávání, tak pod QR kódem je napsán stejný text, jaký je obsažen v QR kódu.

4.4.3 Vytvoření mobilní aplikace klientské části knihovního systému

Pro vytvoření mobilní aplikace je nutné nejprve nainstalovat další moduly.

```
npm install -g cordova
bower install ngCordova
```

Posléze je nutné upravit hlavní skript v klientské části webové aplikace a to tak, že se před samotným načtením AngularJS knihoven musí načíst knihovna ng-cordova.min.js.

```
# app.js
angular.module('bookcase',['ngCordova'])
...
<script src="mobile/ngCordova/dist/ng-cordova.min.js"></script>
<script src="mobile/corodva.js"></script>
...
```

Po této úpravě je nutno říci samotné knihovně pro jakou platformu chceme danou aplikaci vytvořit.

```
cordova platform add [ios/android]
```

V poslední řadě je nutné zaměnit funkci pro čtení QR kódu z námi vytvořeného modulu za funkci implementovanou právě v ngCordova:

```
#controller pro publication_back
```

```
App.controller('qrctrl',['$scope',function($scope,
$cordovaBarcodeScanner)]){
...
$scope.scanbarcode = function(){
$cordovaBarcodeScanner.scan().then(function(imagedata)){
...
console.log(imagedata.text);
...
});
});
...
});
```

Po této uprávě stačí již vygenerovat instalační soubor a nahrát do telefonu. Díky tomu, že bylo při návrhu počítáno s CSS styly i pro mobilní zařízení, pak se nemusejí přidělávat pohledy pro mobilní telefon (platí pro mobilní platformu Android).

4.4.4 Závěr klientské části knihovního systému

Klientská část aplikace byla vytvořena v JavaScriptu při použití frameworku AngularJS. Při psaní klientské části byly vytvořeny moduly pro práci s QR kódy. Stejně jak u serverové části, tak klientská část se dá velmi snadno přesunout a díky stejné možnosti lze pomocí jednoho příkazu nainstalovat veškeré závislosti.

Mobilní aplikace byla vytvořena úpravou webové aplikace a s přídáním CSS stylů pro vzhled, na který je uživatel zvyklý z mobilního telefonu.

5 Závěr

Tato diplomová práce měla několik základních cílů. Prvním z nich bylo seznámení s moderními technologiemi pro vytváření moderních webových aplikací. Bylo provedeno porovnání dvou nejvíce využívaných softwarových sad pro tvorbu moderního webu. Dále bylo provedeno seznámení s potřebami ústavu ITE na knihovní systém. Dle potřeb byl sestaven prototyp databáze.

Dalším cílem diplomové práce bylo vytvoření základního ústavního knihovního systému. Při výběru programovacího jazyka byl brán ohled na multiplatformnost celého systému, proto byl zvolen skriptovací jazyk JavaScript. Kvůli možnosti modularizace a snadné úpravě aplikace pro jiné použití byla zvolená knihovní sada MEAN. Tato sada dala základní souborovou strukturu a logiku knihovního systému. Po naprogramování jádra aplikace bylo navrhnuto API aplikace, které zahrnuje veškeré možné požadavky ze strany klienta na server. Po návrhu API byly naprogramovány stěžejní moduly pro práci, vytvoření QR kódu a jeho zpracování pomocí webkamery nebo fotoaparátu v mobilním zařízení. Následně byly navrženy struktury objektů pro databázi MongoDB. Do jádra aplikace byly objekty doprogramovány tak, aby kontrola dat probíhala jak na straně serveru, tak na straně databáze. Tím vznikl knihovní systém, který lze velmi snadnou úpravou rozšířit o nové funkce nebo ho použít na jiném místě pro jiný účel.

Posledním cílem bylo vytvoření mobilní aplikace. Mobilní aplikace vznikla úpravou webové aplikace. Mobilní aplikaci lze plnohodnotně využívat k vypůjčování a vracení publikací, které se nacházejí v ústavní knihovně.

Během psaní této diplomové práce vznikl základ ústavního knihovního systému, který lze implementovat i na jiné použití. Samotné jádro je taktéž možné přepsat

bez zásahu do klientské části a naopak. Veškeré zdrojové kódy lze nalézt i s návodem pro instalaci pro operační systém Linux na přiloženém CD.

Literatura

[1] *KOSEK, Jiří.***HTML: tvorba dokonalých www stránek : podrobný průvodce.**
Vyd. 1. Praha: Grada, 1998, 291 s. Průvodce (Grada). ISBN 80-7169-608-0.

[2] *KEITH, Jeremy a [foreword by Jeffrey ZELDMAN].***HTML5 for web designers.**
New York: A Book Apart, 2010. ISBN 0984442502.

[3] *KROAH-HARTMAN, Greg.***Linux kernel: in a nutshell. 1st ed.**
Beijing: O´Reilly, 2006, 183 s. ISBN 0-596-10079-5.

[4] *LAURIE, Ben Laurie and Peter.* **Apache: the definitive guide. 2. ed.**
Cambridge [u.a.]: O'Reilly, 1999. ISBN 1565925289.

[5] *DUBOIS, Paul.***MySQL cookbook. 2ND ED.**
England: O'REILLY & ASSOCIATES INC (CA), 2007. ISBN 059652708x.

[6] *TRACHTENBERG, David Sklar and Adam.* **PHP cookbook. 3rd edition.**
Sebastopol, CA: O'Reilly & Associates, 2014. ISBN 9781449363758.

[7] *DIROLF, Kristina Chodorow and Michael a [foreword by Jeremy ZAWODNY].***MongoDB: the definitive guide. 1st ed.**
Beijing: O'Reilly, 2010. ISBN 9781449381561.

[8] *YAAPA, Hage.* **Express web application development: learn how to develop web applications with the Express framework from scratch.**
Birmingham: Packt Pub., 2013, v, 217 p. Community experience distilled.

[9] *LERNER, Ari.***Ng-book: the complete book on AngularJS.**
S.I.: Fullstack.io, 2013. ISBN 099134460x.

[10] *MIKE CANTELON, Marc Harter.***Node.js in action.**
Shelter Island: Manning, 2014. ISBN 1617290572.

[11] *JIM WEBBER, Savas Parastatidis and Ian Robinson.* **REST in practice. 1st ed.**
Farnham: O'Reilly, 2010. ISBN 9780596805821.

[12] *SATRAPA, Pavel.***Perl pro zelenáče: [naučte se programovat v Perlu].**
Praha: Neokortex, c2000, 224 s. Bestseller for all. ISBN 80-86330-02-8.

[13] *PILGRIM, Mark.***Dive into Python 3.**
New York: Apress, c2009, xlix, 360 p. Expert's voice in open source. ISBN 1430224150.

[14] **WAMP server [online].**
2015 [cit. 2015-09-01]. Dostupné z: http://www.wampserver.com

[15] **XAMPP Apache + MySQL + PHP + Perl [online].**
2015 [cit. 2015-09-01]. Dostupné z: https://www.apachefriends.org

[16] **MariaDB [online].**
2015 [cit. 2015-09-02]. Dostupné z: https://mariadb.org/

[17] **Django [online].**
2015 [cit. 2015-09-04]. Dostupné z: http://www.djangoproject.cz/

[18] **Mongoose [online].**
2015 [cit. 2015-09-05]. Dostupné z: http://mongoosejs.com/

[19] **JQuery [online].**
2015 [cit. 2015-09-06]. Dostupné z: https://jquery.com/

[20] **Ab - Apache HTTP server benchmarking tool [online].**
2015 [cit. 2015-09-08]. Dostupné z: http://httpd.apache.org/docs/2.2/programs/ab.html

[21] **Dstat [online].**
2015 [cit. 2015-09-08]. Dostupné z: https://github.com/dagwieers/dstat

[22] *BENADIKOVÁ, Adriana, Štefan MADA a Stanislav WEINLICH.***Čárové kódy: automatická identifikace.**
Praha: Grada, 1994, 252 s. ISBN 80-85623-66-8.

[23] *WATERS, Joe.***QR codes for dummies. Portable ed.**
Hoboken, NJ: John Wiley & Sons, 2012, viii, 112 p. QR codes For dummies. ISBN 1118337034.

[24] *KAŠŠAY, Anton.***Medzinárodné štandardné číslovacie systémy (ISBN, ISSN, ISMN, Bookland EAN) a Slovensko.**
Martin: Matica slovenská, 1993, 20 s. ISBN 80-7090-124-1.

[25] **CentOS [online].**
2015 [cit. 2015-09-09]. Dostupné z: https://www.centos.org

[26] **www.lsoft.cz [online].**
2015 [cit. 2015-09-09]. Dostupné z: https://www.lsoft.cz

[27] **Evilit - Evidence literatury [online].**
2015 [cit. 2015-09-09]. Dostupné z: https://www.normservis.cz/software/evilit

[28] **Technická univerzita v Liberci - Univerzitní knihovna [online].**
2015 [cit. 2015-09-09]. Dostupné z: http://knihovna.tul.cz/